Comparison of Service Selection Algorithms for Grid Services: Multiple Objective Particle Swarm Optimization and Constraint Satisfaction Based Service Selection

Tapashree Guha and Simone A. Ludwig
Department of Computer Science
University of Saskatchewan
Canada
{t.guha,ludwig}@cs.usask.ca

Abstract

Grid computing has emerged as a global platform to support organizations for coordinated sharing of distributed data, applications, and processes. Furthermore, Grid computing has also leveraged web services to define standard interfaces for grid services adopting the service-oriented view. Consequently, there have been significant efforts to enable applications capable of tackling computationally intensive problems as services on the Grid. In order to ensure that the available services are optimally assigned to the high volume of incoming requests, it is important to have an efficient service selection algorithm. The algorithm should not only increase access to the distributed services, promoting operational flexibility and collaboration, but should also allow service providers to scale efficiently to meet a variety of demands while adhering to certain current quality of service standards. This paper, proposes and compares two service selection algorithms on the Grid: the Multiple Objective Particle Swarm Optimization algorithm using Crowding Distance technique (MOPSO-CD) to the Constraint Satisfaction based Matchmaking (CS-MM) algorithm.

1. Introduction

With the increase in complexity and scale of domain problems in science and engineering, the last few decades have seen a proportionate increase for the need of high performance applications to solve and model these problems. For example, in research areas like fluid dynamics, molecular dynamics, global climate modeling, etc., researchers are depending on high performance applications to explore and simulate interesting phenomena in these fields [14]. Additionally, due to the popularity of high-speed networks, and the World Wide Web, there has been a significant effort for providing these high performance applications as Web and Grid services. These services are accessed remotely across the network by the users, which not only promote seamless execution of the jobs submitted by the users, but also enable sharing of these services amongst various users. Few examples of high performance services that are available online are: data mining, theorem proving and logic, parallel numerical computations, etc. Moreover, for the high-demand services, it will often be the case that the services are replicated at multiple sites so that in a situation where there is a large number of requests for the services, all the service providers together can serve the client requests. Therefore, in order to tackle the high volume of user requests for these services dynamically, a high-performance computational resources is needed, to achieve the goals of executing the client requests remotely, and efficiently.

The Grid, at the same time has rapidly emerged as the dominant paradigm for wide area high-performance distributed computing. The basic purpose of the Grid is to provide a service-oriented infrastructure that uses standardized protocols, allow users to have seamless access, and the coordinated sharing of geographically distributed computing resources providing various services. Also, as the Grid computing paradigm is being rapidly deployed by both industry and researchers, it is enabling a new generation of high performance applications to be developed that can be seamlessly accessed by the users.

However, before utilizing these services to execute a client request, a requester needs to select an appropriate service provider from a service provider pool deployed on the Grid. Once a service provider is selected, the site will assign the appropriate number of resources to the request. Hence it is seen that, these services necessitate an effective coordination of participating sites to handle the request(s). Most importantly it gives rise to the need of a faster and accurate selection process in order to accommodate the execution of a wide spectrum of client requests. Otherwise, there might be discrepancies in the way the requests are being served.

Furthermore, as mentioned in [9], for a faster and accurate selection of service providers it is required, that the service selection procedure complies with a particular Quality of Service (QoS) metric as not meeting such criteria can impose a significant impact on the accuracy of the results. The QoS metric used should not only consider the characteristics of the network linking all these service provider sites, but should also be designed in the context of Grid services. For efficient and satisfactory services, a user requesting a particular service must identify a set of QoS requirements, such as the bandwidth of the network, number of processors, the time duration for which the service is required, and the expected cost the user is willing to pay, etc. At the same time, a service provider also has its capability listed in the form of a QoS metric. Among this QoS metric, some are decided by the service providers themselves, such as the cost, while others are provided by third party and network managers [1].

For an efficient service selection method, it is necessary that the algorithm is able to automatically discover services and choose between a set of similar services. The algorithm should be capable of handling two primary scenarios: 1) be able to handle multiple client requests coming in at the same time and 2) be able to handle situations where more than one client is requesting a similar service. To incorporate the second scenario, a QoS metric including parameters such as availability, reliability, etc. of the service providers are essential properties to consider, guaranteeing an efficient service selection on the Grid.

This paper presents a swarm intelligence based approach to select services on the Grid based on a QoS metric using the Multiple Objective Particle Swarm Optimization algorithm with Crowding Distance technique (MOPSO-CD) to find the best match between the client requests and the service providers. This algorithm is compared with an enhanced version of a classical matchmaking algorithm known as the constraint satisfaction based matchmaking algorithm (CS-MM); and experiments are performed and results of the same are presented.

The rest of the paper is structured as follows: In the next section related work of this research is presented, Section 3 describes the approach chosen for the experiments, i.e., it discusses the QoS metrics used, the CS-MM algorithm, and the MOPSO-CD algorithm. In Section 4 the experiment setup and the results are discussed, after comparing both algorithms. Finally, in Section 5, the conclusion is presented.

2. Related Work

Service computings main goal is the provisioning of services, such as services for high performance computing. As the scope of high performance applications being deployed as services on the Grid becomes wider, the need for an efficient and sophisticated approach to deploy the service computing infrastructure becomes even more evident. The service selection is an important issue for the Grid as it defines the process for locating service providers and retrieving service descriptions [7]. The most prominent service discovery method as discussed in the literature is the semantic matchmaker [6]. Essentially, the semantic matchmaker method proposes a solution for service discovery on the Grid. This framework is based on three selection stages which are context, semantic and registry selection. Instead of only performing service name matches which other common service discovery systems are restricted to, the semantic matchmaking framework provides a better service discovery process by using service semantics stored in ontologies. The framework permits Grid applications to specify the criteria with their service request. These criteria are then matched with those of the services available, and hence enable interoperability in the matchmaking process [8, 9]. This matchmaking process calls for optimization, especially when large numbers of requesters are competing for similar kind of services.

A significant amount of research has been conducted in the area of service selection on the Grid. However, much of the work is based on the accurate predictions of the completion times of jobs when selecting services for the jobs. A scheduling algorithm that is driven by a user supplied application deadline and a resource access budget is proposed in [2]. The algorithm selects services in such as way that user requirements are met (e.g. duration of execution), and yet it keeps the cost of computation at the minimum. The work in [11] provides a service selection in which, for each user request, a scheduler selects a set of servers that can handle the computation and ranks them based on the minimum completion time. On the other hand, [5] presents a general-purpose service selection framework that supports both single-service and multiple-service selection on the Grid. Their work extends the Condor matchmaking framework. All the above service selection methods specifically concentrate on one main attribute i.e., completion time. Intuitively, it can be claimed that, by allowing users to specify more characteristics of resources to be used, the flexibility and usability of the entire service selection process can be increased profoundly.

A classical matchmaking algorithm, as discussed in [9] implements the service selection methods, based on five generic QoS criteria for elementary services. The match score for a requester-service pair is calculated based on these five criteria. The algorithm considers the first request from the list of requesters, scans the list of the services discovered, and assigns the service with the highest match score to the request. The selected service is then removed from further consideration. Similarly, the best match score for the remaining services is determined for the second requester; then the corresponding service is removed from the list of the services available. This goes on until each client request is matched with a service.

The advantage of this approach is that each service requester is considered only once and hence avoids redundancy in the service allocation process; while the disadvantage lies in the fact that the client requesters later in the list are very likely to get assigned to a service with a worse match score. And the likelihood of a client request being assigned to poor service providers increases with the increase in the number of requesters accessing similar kind of services.

To overcome this disadvantage initially an enhanced version of the matchmaking algorithm named as the CS-MM algorithm is proposed. In this algorithm the requesters are matched with a service after considering the entire list of service providers, instead of considering it sequentially. The problem of assigning the requesters appearing later in the list to the inefficient services, was solved considerably by changing the method of choosing the best matched service from the service provider list. However, in order to make the optimization process more efficient, and to produce better results under the circumstances where large numbers of requesters are competing for similar kind of services, the swarm intelligence based MOPSO-CD algorithm was used.

The aim of this research work is to find an algorithm which enables optimal selection of services on the Grid while considering the following scenarios: 1) multiple client requests have to be satisfied at the same time, 2) more than one client can request a similar service, 3) the delivery of a service for each request should be guaranteed, and 4) the matching of client requests with service providers should be optimized in order to avoid matching the requests later in the list with a worse match score. This paper, proposes two different approaches used to achieve the above goal, shows experiments performed, and compares the results.

3. Proposed Approaches

The problem of service selection on the Grid consists of finding an efficient algorithm which can match multiple client requests and service providers efficiently, while optimizing the multiple objectives (the 5 QoS parameters) at the same time. The problem at hand also includes: handling more than one client request at the same time, satisfying multiple clients requesting similar services, guarantying a service for each request, and finally optimizing the assignment process between the service requesters and the service providers, so that none of the requests are compromised.

In this paper, two service selection algorithms for the Grid, are proposed and compared: 1) the MOPSO-CD algorithm and, 2) the CS-MM algorithm. Based on the assumption that both the service requesters and the service providers have certain QoS parameters, the algorithms implemented finds the requester and service provider pairs with the highest match score by using optimization techniques.

The design of the approach taken to achieve the above goal includes three primary steps: 1) Defining the QoS metric for each requester and service, 2) Implementing the CS-MM algorithm, and 3) Implementing the MOPSO-CD algorithm.

3.1. QoS Metric

The QoS metric plays an important role for the service selection on the Grid. It is especially helpful when there is more than one similar service provider available for a particular client request. In this situation the QoS metric helps to allocate the request to the most qualified service. The five generic quality of service criteria considered in this work are the same as in [9]. These QoS metric can be adopted by all Grid and Web services. Although only five generic QoS parameters have been considered for illustrating and comparing the algorithms, new criteria can be added without necessarily altering the service selection algorithm. In this investigation, the quality criteria in the context of services are: *(1) execution price, (2) execution duration, (3) reliability, (4) reputation, and (5) availability.* The values of these QoS parameters range from 0 to 1. Each requester provides the QoS value of based on its requirement of how the request must be executed, and each service provides the value based on its capability to execute the task.

3.2. Constraint Satisfaction based Matchmaking (CS-MM) Algorithm

In this research an assumption has been made that each service and requester, have five QoS parameters, such as service S_i and requester R_j have the following QoS metric $[S_{i1}, S_{i2}, ..., S_{i5}]$ and $[R_{j1}, R_{j2}, ..., R_{j5}]$ respectively. It is important to note that the values of these QoS attributes range from 0 to 1. The five attributes of each service and requester are recorded in two 2-dimensional matrices. The orders of these matrices are $n \times 5$, and $m \times 5$, where *n* is the number of requester and *m* is the number of services.

The first step of this algorithm is to calculate the match value for each QoS parameter of each requester-service pair. The match value represents the distance between QoS values of the requester-service pairs. The match value $MV_{(R_{jx},S_{ix})}$ for the x^{th} QoS metric for R_j and S_i requester-service pair is calculated as follows:

$$MV_{(R_{jx},S_{ix})} = 1 - (\frac{|R_{jx}-S_{ix}|}{R_{jx}}) \text{ Eq (1)}$$

Whereby the value of *i* and *j*, ranges from 1 to *n* and *m* respectively, where *n* and *m* are the total number of services and requesters at a certain time; *x* represents the QoS parameter; R_{jx} : represents the service requester R_j value for the x^{th} QoS parameter; S_{ix}: represents the service provider S_i value for the x^{th} QoS parameter. The match value is subtracted from 1, which means, the higher the value of $MV_{R_{jx},S_{ix}}$ the better is the match. The overall match score $MS_{(R_j,S_i)}$ for the requester-service pair R_j and S_i is calculated as follows:

$$MS_{(R_j,S_i)} = \frac{\sum_{x=1}^{n} MV_{(R_{jx},S_{ix})}}{n} \text{ Eq (2)}$$

Whereby the value of *n* is the number of QoS parameter considered for the problem. In this case the value of *n* is set to five. The CS-MM algorithm calculates the match score for each requester-provider pair using equations (1) and (2). It stores the match score value for each requester-service pair in a MS matrix as follows:

$$\mathbf{MS} = \begin{pmatrix} MS_{11} & MS_{12} & . & . & . & MS_{1m} \\ MS_{21} & MS_{22} & . & . & . & MS_{2m} \\ . & . & . & . & . & . \\ \\ MS_{n1} & MS_{n2} & . & . & . & MS_{nm} \end{pmatrix} \text{Eq(3)}$$

Whereby the value of MS_{nm} in the above matrix is the match score of the n^{th} requester and the m^{th} service. Once the matrix is formed, the highest value from each row is selected. This value represents the best possible match of the corresponding requester and service. In the event, where more than one requester is assigned to a single service, constraint satisfaction problem algorithm [10] approach is applied. The constraint in this problem is to assign a single requester to any service, without affecting the overall match score of the requester-service pair significantly. This constraint is based on the assumption that a service can only serve one requester at a certain time. And hence, in order to satisfy this constraint the second highest match score for the service-requester pair is considered, and is subtracted from the highest value. The requester-service pair with the minimum difference between these values is selected. This process continues until each service is matched with a single requester.

The strength of the algorithm lies in the fact that it matches requesters with services in a fair fashion while optimizing the match score of each pair and it also avoids the problem seen in the classical matchmaking algorithm, i.e., assigning the requesters later in the list with remaining services, resulting in an inferior match score.

3.3. Multi Objective based Particle Swarm Optimization Algorithm using Crowding Distance (MOPSO-CD)

The MOPSO-CD algorithm is a swarm based artificial intelligence algorithm, which takes inspiration from the social behavior of bird flocking or fish schooling behavior as seen in nature. More about multi-objective particle swarm optimization algorithms can be found in [3, 4, 12]. A brief description of the MOPSO-CD algorithm used to optimize the service selection methods on the Grid is given in this section, however a more detailed description of this algorithm can be found in [13].

Algorithm 1: MOPSO-CD Algorithm

```
Data: List of Requesters and Services.
Result: Non-dominated requester-service pairs.
begin
    $t \longleftarrow 0$ (Initialize iteration counter)
    for $1 \leq i \leq n$ do
        Randomly initialize P[i]
        Initialize V[i] =0
    for $1 \leq i \leq n$ do
        Evaluate P[i]
        Set Pbest of P[i]
        Set Gbest of P[n]
    while $P[i] \neq 0$ do
        Compare each solution for dominance
        $A \longleftarrow NDS$
        (NDS = non-dominated solutions in P[i])
    while !max iterations do
        Calculate crowding distance (CD) for P[i]
        $A \longleftarrow SortAinDescendingorderofCD$
        for $1 \leq i \leq n$ do
            Set Gbest in P[n]
            Update matchscore for P[i]
            if $P[i] = NDS$ then
                $A \longleftarrow NDS$
                Set Pbest of P[i]
        $t \longleftarrow t + 1$
end
```

The first step of the algorithm is to randomly initialize all the particles in the swarm, with the list of services and requesters along with their five QoS parameters. The random initialization of this problem refers to randomly matching the requesters and the services, and generating a match score for the same. This step also includes initializing the local best, i.e., the highest match score value amongst the neighboring particles of each particle in the swarm, and also finding the global best, i.e., the best match score value amongst all particles in the swarm, for the current iteration.

The match score in this case is calculated using equations (1) and (2) mentioned in the previous section. In classical particle swarm optimization terms, these match scores are referred to as the (potential) solutions, and each particle in the swarm represents one of the potential solutions to the problem.

In the next step the non-dominated solutions, i.e., the solutions that are not dominating any other solution in the list of the potential solutions are found and are stored in an archive. The crowding distance values of each non-dominated solution in the archive list are calculated, and are sorted according to the descending order of the crowding distance.

After the list has been sorted, for each non-dominated solution the new match scores are computed based on the value of the global best particle of the iteration. The match score of each requester-service pair is updated to a higher value by matching it with a better service (where possible), whereby optimizing the whole assignment process. This continues until the maximum number of iterations has been reached. Unlike as mentioned in [13], no mutation operators has been included in this algorithm, as the QoS metric of requesters and services are fixed, and cannot change over a period of time. A brief pseudo code of the algorithm is given in Algorithm 1.

4. Experiments and Results

The results presented in this section, reflects the efficiency of both algorithms to assign requesters to services based on their QoS metric. The comparison of these algorithms is based on two sets of measurements: 1) the average fitness score achieved for the given service requester and service provider set, and 2) the execution time of both algorithms. The experiments were run on an DELL INSPIRON 6400 laptop, with Intel Core Duo Processor, 1 GB RAM, and 100 GB HDD. The measurements were repeated 10 times and the average value was taken for each point.

Figure 1 plots the variance of the average fitness score versus the number of iterations, keeping the requester-service pairs fixed at 50-50, 100-50, and 50-100 respectively for the MOPSO-CD algorithm. From this graph it can be concluded that the algorithm performs efficiently by achieving an average fitness scores of 0.89, 0.75, and 0.96 at 3500 iterations for 50-50, 100-50, and 50-100 requester-service pairs respectively. It can also be concluded that the increase of the global best values in the initial iterations is more rapid than in the later ones, and that the value of the global best for all three cases tends to stabilize after 3000 iterations.

Figure 1 also shows a similarity between the converging behaviors of the 50-50 and 50-100 requester-provider cases. Initially, in both the scenarios the average fitness score os-

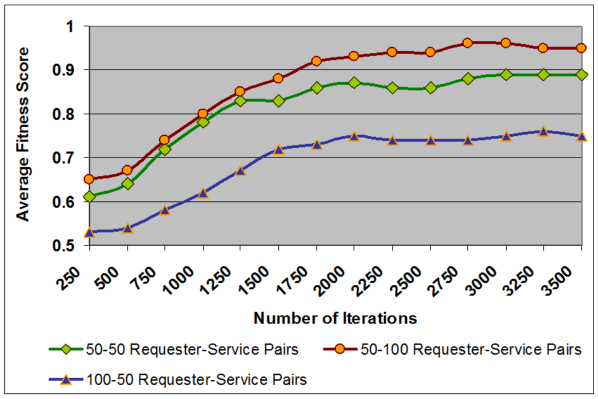

Figure 1. Average fitness score vs. number of iterations of MOPSO-CD algorithm.

cillates between very close ranges of approximately 0.6 and 0.85. However, approximately after 1300 iterations the 50-100 requester-provider case, starts outperforming the 50-50 case. The reason why the average fitness in case of the 50-100 requester-service pair is higher than the 50-50 one, is explained in the next few paragraphs.

Figure 2 on the other hand reflects the average values of the fitness score for 50-50, 100-50, and 50-100 requester-service pairs for the CS-MM algorithm. From this graph it can be concluded that the algorithm achieves a fitness score of 0.74, 0.62 and 0.93 for 50-50, 100-50, and 50-100 requester-service pairs respectively. Comparing the aver-

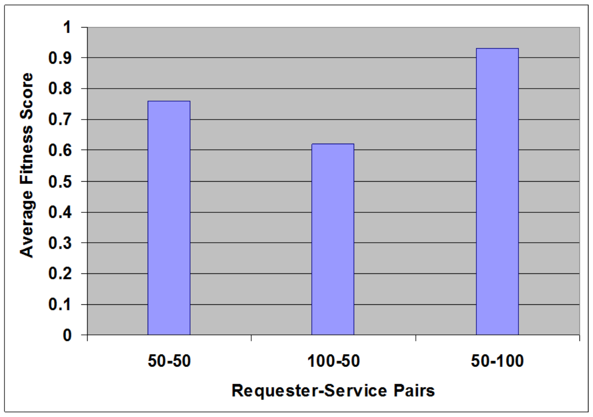

Figure 2. Average fitness score for requesterservice pairs of CS-MM algorithm.

age fitness scores achieved by both algorithms it is seen that the MOPSO-CD algorithm achieves a better average fitness score than the CS-MM algorithm. A graphical representation of this comparison is shown in Figure 3. From the graph it can be inferred that the MOPSO-CD outperforms the CS-MM algorithm significantly in the case of 50-50 and 100-50 requester-service pairs, whereas the average fitness scores achieved for the 50-100 requester-service pairs are comparable for both algorithms.

It is important to note that for the above measurements the number of iterations for the MOPSO-CD algorithm was fixed to 3000, as it can be seen from Figure 1, that the average fitness scores follows a stable trend after that. A more detailed comparison of the average fitness scores achieved by both algorithms is presented in a tabular format in Table 1. It is seen that for MOPSO-CD the average fitness scores for 50-50, 100-50, and 50-100 requester-service pairs stabilizes at approximately 0.89, 0.96 and 0.75 respectively for 3000 iterations, while the average fitness scores for CS-MM algorithm for the three scenarios is 0.74, 0.62 and 0.93 respectively.

Furthermore, the following conclusions can be drawn from the results obtained. Firstly, the average fitness scores of 0.93 and 0.96 for the CS-MM algorithm and the MOPSO-CD algorithm respectively confirms that the average fitness scores achieved during the 50-100 requester-service pair scenario is the highest, compared to the other two scenarios of 50-50 and 100-50 requester-service pairs. This can be explained on the basis that, in the scenario where there are more services available for a set of requesters, each requester has an option of getting its job assigned to a service choosing from a larger pool of services. In this case, each requester can choose one service provider from a set of two service providers. In this situation, a requester is more likely to get assigned to a service provider with the desired QoS metric, and hence the average fitness score achieved is higher for both algorithms compared to the other two cases.

Secondly, looking at the percentage increase in the average fitness score values of the MOPSO-CD algorithm in comparison with the CS-MM algorithm, a conclusion can be drawn that, in the 50-100 requester-service pair scenario both the average fitness scores are comparable. In this case the increase in the average fitness score using the MOPSO-CD algorithm in comparison to the CS-MM algorithm is only 3.23%. Furthermore, in the situation of 50 services for 100 requesters, although both algorithms could match only 50 requesters at a time (as the experiments considers only static allocations of the requests) the average fitness scores achieved by the MOPSO-CD is approx. 27.41 % better than that of the CS-MM algorithm. This scenario is essentially similar to cases where 50 requester and 50 services pairs are optimized. It can be seen that the MOPSO-CD outperforms the CS-MM algorithm in a significant way by 20.27 %. Amongst the three scenarios, these are the two scenario in which maximum optimization needs to be done, as there are equal numbers of services available for each single requester and hence the competition between the requesters to assign their jobs to a service is very high. The assignment process also becomes difficult during these two scenarios because more than one client might have requested a similar kind of service, which makes the optimization process a necessity, in order to satisfy all the requesters unbiasedly.

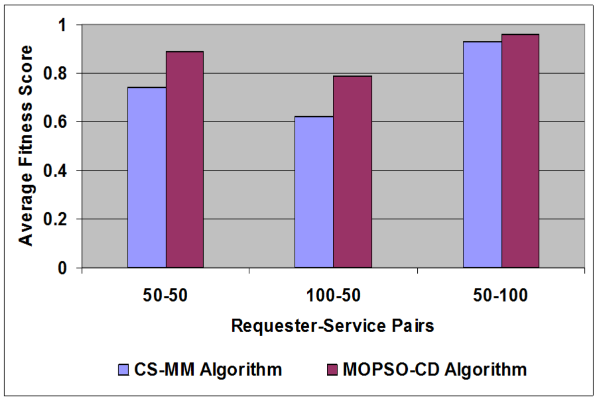

Figure 3. Average fitness score for requesterservice pairs of CS-MM and MOPSO-CD algorithm (at 3000 iterations).

Table 1. Comparison of fitness score

Requester- Service pairs	Fitness Score	
	CS-MM	**MOPSO-CD**
50-50	0.74	0.89
100-50	0.62	0.79
50-100	0.93	0.96

Also, considering that the highest value of the fitness can only be 1, the experimental results confirm that the average fitness score achieved for all three scenarios, in case of MOPSO-CD at 3000 iterations is 0.88, while the average fitness score value achieved by the CS-MM algorithm is 0.76, which is approximately 15.6 % less than the former one. These results strongly support the hypothesis that using the MOPSO-CD algorithm to optimize the process of assigning requesters to services can achieve high accuracy in terms of better and unbiased matches, in comparison to the CS-MM algorithm.

The next set of experiments compare the execution time

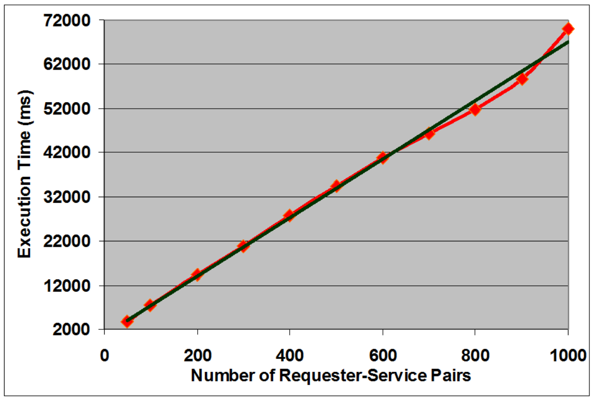

Figure 4. Execution time (ms) vs. number of requesters-service pairs of MOPSO-CD algorithm.

of the two algorithms. Figure 4 shows the variance of execution time in milliseconds versus the different number of requester-service pairs, for the MOPSO algorithm, keeping the maximum number of iterations fixed to 3500, while Figure 5 shows the variance of the execution time for the CS-MM algorithm. The reason behind choosing 3500 iterations as a benchmark is that after 3000 iterations the particles starts converging. The number of requester-services pairs was varied from 50 requester-service pairs to 1000 requester-service pairs, in order to get a fair distribution.

In case of the MOPSO-CD algorithm, 100, 500 and 1000 requester-service pairs resulted in execution times of 7359, 34282 and 70094 ms respectively, while for the CS-MM algorithm the execution times for the same requester-service pairs resulted in 169, 395 and 654 ms respectively.

The equation to calculate the execution time for the MOPSO-CD algorithm and the CS-MM algorithm is $y = 66.265x + 748.15 = [ms]$ and $y = 0.5424x + 111.06 = [ms]$ respectively. Table 2 illustrates the comparison of the execution time in ms for 200, 400, 600, 800, 1000 requester-service pairs. From Figure 4 and 5 it can be concluded that with the increase in the number of the provider-requester pairs, the execution time increases linearly. From Table 2 it can inferred that CS-MM is faster than the MOPSO-CD algorithm. And, the execution time comparison with the number of requesters for both the MOPSO and the CS-MM algorithms has a linear distribution. This significant difference in speed of the MOPSO-CD and the CS-MM algorithm can be explained on the basis, that the MOPSO-CD algorithm follows the rules of the swarm intelligence algorithm, versus the fact that CS-MM uses simple calculations.

Explaining further, in case of MOPSO-CD algorithm the problem entities (in this case it is the requester-provider pairs) are represented as swarms. And the total execution time of the of this algorithm can be attributed to the fact that, each swarm migrates around the solution space keeping a track of its own as well as its neighbors fitness score, updates its own fitness score by a factor based on its knowledge about the global best swarm for the current iteration - to find the global optimal solution, while the CS-MM algorithm matches the requester-service pairs based on a series of simple algebraic and arithmetic calculations, which is not very computationally intensive.

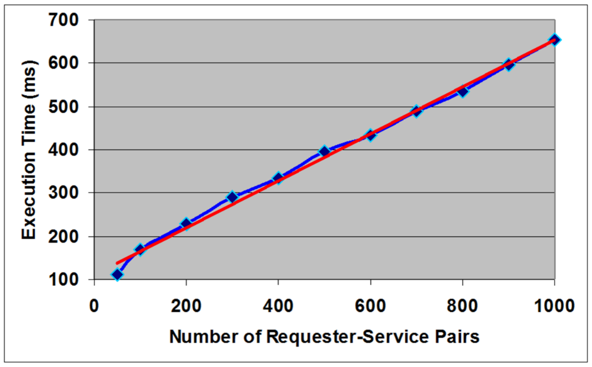

Figure 5. Execution time (ms) vs. number of requester-service pairs of CS-MM algorithm.

Table 2. Percentage difference of execution time

Requester- Service pairs	Execution time (ms)	
	MOPSO-CD	CS-MM
100-100	14422	229
200-200	27813	334
300-300	40812	432
400-400	51640	534
500-500	70094	654

Looking at the results, it can be concluded that even though the MOPSO-CD takes longer than the CS-MM algorithm to compute, it matches the provider-service pairs with a higher fitness score hence the match process is more optimized, avoiding any biased matches.

5. Conclusions

The Grid is a high performance computing paradigm, which is rapidly emerging as a suitable platform to deploy distributed applications. The feasibility of this type of supercomputing has already been demonstrated by the large number of applications deployed on the Grid. To increase the effectiveness and applicability of these high performance applications on the Grid, having an efficient service selection algorithm has become mandatory. Service selection is one of the basic components of a Grid system, to discover qualified service providers for different Grid users. A service selection algorithm should be efficient to ensure unbiased allocations of requesters to the service providers. To achieve efficiency, the service selection algorithm should match the services and requesters in a way which optimizes the entire allocation process.

This paper compared the Multiple Objective Particle Swarm Optimization algorithm using Crowding Distance technique (MOPSO-CD) to the Constraint Satisfaction based Matchmaking (CS-MM) algorithm. It showed that using the MOPSO-CD algorithm for selecting services deployed on the Grid for several requesters accessing it achieves a higher match score than using the matchmaking algorithm. The reason behind the efficiency of the MOPSO-CD algorithm can be attributed to the three key features of the algorithm. Firstly, the algorithm uses an external repository or archive of non-dominated solutions found in previous iterations and hence it can keep track of all the non-dominated solutions of previous generations. Secondly, because it uses the crowded comparison operator, which basically computes the average distances of each solution between the neighboring solutions, it acts as a diversity operator and diversifies the search in the problem space. And thirdly, the particles in the swarm keep track of the local best solution and global best solution, and use this information to find the optimal match score. These characteristics of the MOPSO-CD, along with the experimentation results verifies, that the MOPSO-CD algorithm, which is a swarm intelligence based optimization technique, proves to be a powerful and accurate method to handle multi-objective optimization problems such as the one of service allocation.

After analyzing the experiment results of both the approaches it is recommended that if the execution time plays an important role, then the CS-MM approach of service selection should be used. However, if the accuracy is paramount, the MOPSO-CD approach outperforms CS-MM by a wide margin and therefore should be chosen.

6. Acknowledgement

The authors wish to acknowledge NSERC (Natural Sciences and Engineering Research Council), for partly funding this research work.

References

[1] R. Al-Ali, O. Rana, D. Walker, S. Jha, and S. Sohail. G-QoSM: Grid Service Discovery Using QoS Properties. *Special Issue on Grid Computing, Computing and Informatics Journal*, 21(4):363–382, 2002.

[2] R. Buyya and M. Murshed. A Deadline and Budget Constrained Cost-Time Optimisation Algorithm for Scheduling Task Farming Applications on Global Grids. *Arxiv preprint cs/0203020*, 2002.

[3] C. Coello, C. Lechuga, S. de Computacion, and M. CINVESTAV-IPN. MOPSO: a proposal for multiple objective particle swarmoptimization. *Proceedings of the 2002 Congress on Evolutionary Computation, 2002. CEC'02.*, 2, 2002.

[4] X. Hu and R. Eberhart. Multiobjective optimization using dynamic neighborhood particleswarm optimization. *Proceedings of the 2002 Congress on Evolutionary Computation, 2002. CEC'02.*, 2, 2002.

[5] C. Liu, L. Yang, I. Foster, and D. Angulo. Design and evaluation of a resource selection framework for Grid applications. *Proceedings of 11th IEEE International Symposium on High Performance Distributed Computing, 2002. HPDC-11 2002.*, pages 63–72, 2002.

[6] S. Ludwig and S. Reyhani. Introduction of semantic matchmaking to grid computing. *Journal of Parallel and Distributed Computing*, 65(12):1533–1541, 2005.

[7] S. Ludwig and S. Reyhani. Semantic approach to service discovery in a Grid environment. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(1):1–13, 2006.

[8] S. Ludwig and P. van Santen. A Grid Service Discovery Matchmaker based on Ontology Description. *EuroWeb 2002*.

[9] S. A. Ludwig and S. Reyhani. Selection algorithm for grid services based on a quality of service metric. *21st International Symposium on High Performance Computing Systems and Applications, 2007. HPCS 2007.*, pages 13–13, May 2007.

[10] B. Nadel. Constraint satisfaction algorithms 1. *Computational Intelligence*, 5(3):188–224, 1989.

[11] H. Nakada, M. Sato, and S. Sekiguchi. Design and implementations of Ninf: towards a global computing infrastructure. *Future Generation Computer Systems*, 15(5-6):649–658, 1999.

[12] K. Parsopoulos and M. Vrahatis. Particle swarm optimization method in multiobjective problems. *Proceedings of the 2002 ACM symposium on Applied computing*, pages 603–607, 2002.

[13] C. Raquel and P. Naval Jr. An effective use of crowding distance in multiobjective particle swarm optimization. *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 257–264, 2005.

[14] J. Weissman. Adaptive Resource Selection for Grid-Enabled Network Services. *Proceedings of the Second IEEE International Symposium on Network Computing and Applications*, 2003.